ECE 484:安全自主原则 秋季2024 *HW1, MP1:车道检测*

作者:ECE484团队

截止日期:2024年9月27日

1 介绍

在这个作业中，你将应用计算机视觉技术进行车道检测。在第一部分中，你将回答4个与计算机视觉概念相关的书面问题。然后，你将实现一个基于摄像头的车道检测模块。该模块将以自动系统中使用的格式视频流作为输入，并将生成标记车道区域的标注视频。我们将视频分割成帧，并将其作为图像在ROS(机器人操作系统)中处理。您将依次完成图5所示的功能，以创建车道检测模块。

对于第一部分，您将需要解决4个书面问题。对于第二部分，你所有的代码都应该在文件studentVision.py和line_fit.py中，你必须写一份简短的报告mp1_<groupname>.pdf。您将不得不使用机器人操作系统(ROS)[2]中的函数，并鼓励您使用OpenCV库[1]。本文档为您提供了开始实现车道检测模块的第一步。你必须利用网上可用的教程和文档。在你的报告中引用所有资源。学生守则中列出的所有关于学术诚信和抄袭的规定都适用。

学习目标

- 使用梯度进行边缘检测
- 使用rosbags, Gazebo和Rviz
- 基本的计算机视觉
- 使用OpenCV库

系统需求

- Ubuntu 20.04
- ROS Noetic

2 作业问题

*注意，对于证明，我们希望你自己写证明，而不是简单地从互联网上复制答案。自己证明这些是测验/期中考试的好练习。顺便说一句，当学生逐字抄写校样时，通常是很容易确定的。我们很乐意在办公时间、家庭作业派对或校园网络中提供帮助。

问题1卷积[个人](15分)自己做。单位冲量是卷积的恒等性质。然而，如果给定一个函数(或数组)f和常数k, f*kδ = kf是否成立?如果是，请证明这个性质。否则，提供一个反例来说明为什么它不成立。你可以认为函数是一维的。请展示所有的步骤，并在必要时给出证明。

问题2卷积[个体](20分)卷积有很多性质。证明分配律成立。即，给定函数f、g和h，表明h*(f+g) = h*f+ h*g。同样，你可以认为这些函数是一维的。请展示所有的步骤，并在必要时进行说明。

问题3高斯函数[个体**](15分)**证明一个二维高斯函数$G_\sigma(x, y)$可以被分割成两个函数的乘积。请展示所有步骤，必要时进行证明。一个应该是函数x，另一个是函数y。为什么可分离性的存在很重要?

问题4中值滤波器[个体](20分)在讲座中，你学习了滤波器和卷积。有许多类型的过滤器，包括高斯，索贝尔(你将在这个MP中使用)和中值过滤器。回想一下，一个LSI滤波器(其中线性和平移不变性保持的属性)可以表示为卷积。讲座中主要讨论的是线性滤波器。如你所料，非线性滤波器也存在。图1提供了一个中值滤波器应用于小图像的例子作为参考。

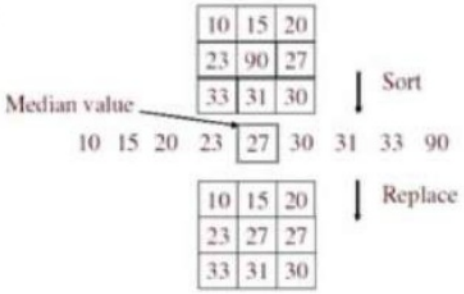

图1:中值滤波函数来源K. Grauman

中值滤波是线性的还是非线性的?请解释原因。提供一个2D的例子(不是从笔记、幻灯片、课程阅读材料中)。注意这个例子应该正式地说明线性是否成立。你不需要写证明，但它必须植根于数学(例如，从线性的定义出发)。

问题5 OpenCV中的**滤波[个人](30分)**在课堂上，我们通过声称它们可以帮助去除嵌入在图像中的各种类型的噪声，来激发将滤波器应用于图像、脑电图数据等的有用性。噪声有很多种类型，包括高斯噪声、泊松噪声、斑点噪声、盐胡椒噪声等。在mp1/src文件夹下，你会发现一个名为filter_main.py。实现filter_gaussian和filter_median滤波器函数，并测试它们在椒盐噪声和高斯(白)噪声上的性能。哪一种滤波器更适合滤除椒盐噪声?哪种滤波器更适合滤除高斯(白)噪声?在报告中附上你的结果图像，并解释为什么你认为特定的滤波器对某些噪声效果更好?(你需要做源开发/设置。Bash在这一步。详情请参考后面章节)

图2:(a)盐胡椒噪声(b)高斯(白)噪声

3 实现

3.1 问题陈述

对于实现，您将实现一个车道检测算法，将应用于2个场景。在场景1中，你会看到一段在高速公路上行驶的真实汽车上录制的视频。

图3:在真车上录制的视频，分辨率1242 ×375

在场景2中，您将在Gazebo中使用配备相机的GEM汽车模型。GEM车将沿着轨道行驶。

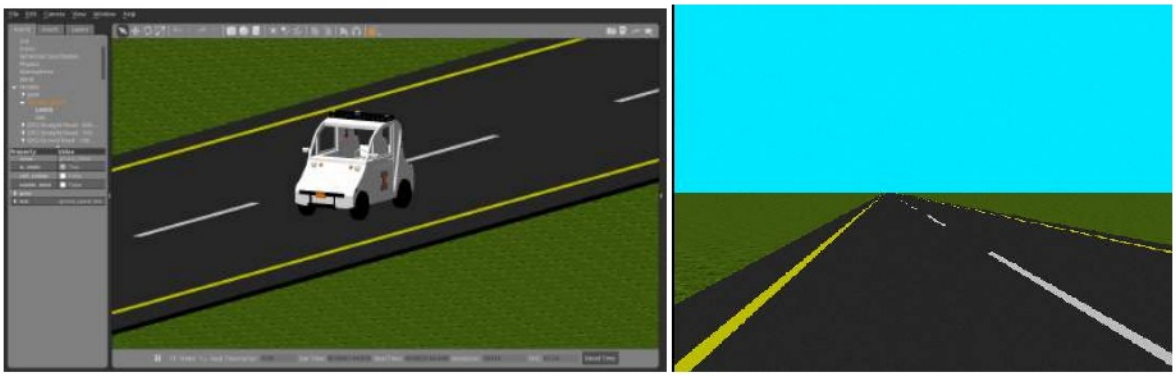

图4:GEM汽车及其在Gazebo中的摄像头视图

对于这两种情况，你的任务是使用你的算法来检测视频或摄像机输出中的车道线。请注意，两个场景中的几个参数(例如颜色通道，索贝尔滤波器的阈值，透视变换的点)是不同的，您需要分别计算它们。

3.2 模块架构

一个典型的车道检测管道的模块功能如图5所示，并在下面列出。然而，在这个MP中，你只需要实现一些功能。标有*的功能不是你必须实现的，但你可以尝试一下。车道检测是一个难题，有很多正在进行的研究。这可能是一个组件，以在您的项目中进行更深入的探索。

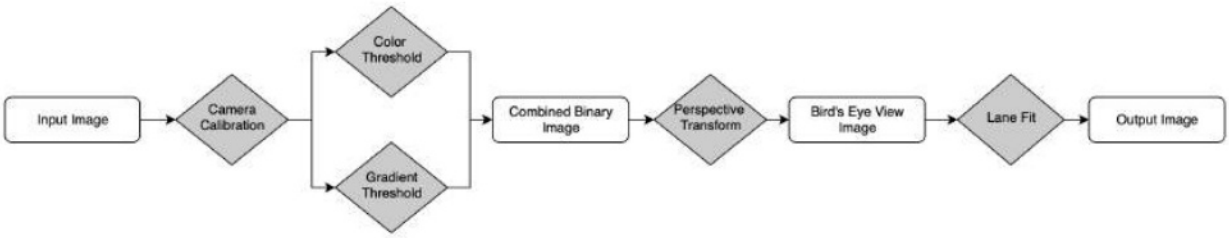

图5:车道检测模块概述。

相机校准*由于所有相机镜头都会在图像中引入一定程度的畸变，因此在我们开始处理图像之前，需要进行相机校准。

透视变换将图像转换成*鸟瞰图*。

颜色阈值颜色通道上的阈值，以找到车道像素。一个典型的方法是将图像从RGB空间隐蔽到HLS空间，并对S通道进行阈值化。还有其他的方法来完成这一部分。

梯度阈值通过应用索贝尔滤波器对图像进行边缘检测。

组合二值**图像**将梯度阈值和颜色阈值图像相结合得到车道图像。建议应用*感兴趣区域*掩模去除无关背景像素，应用形态学函数去除噪声。

从二值*眼视图*图像中提取*左右*车道*中心*的坐标。将坐标拟合成两个表示左右车道的二阶多项式。

4 开发指令

你需要修改的所有python脚本是studentVision.py和line_fit.py，它们位于以下路径:

```
cd [path-to-MP1-workspace]/src/mp1/src
```

4.1 梯度阈值

在本节中，我们将实现一个函数，该函数使用梯度阈值来检测图像中有趣的特征(例如车道线)。输入图像来自回调函数。输出应该是一个突出边缘的二值图像。

```
def gradient_threshold(self, img, thresh_min= 25, thresh_max= 100):
    """
    Apply sobel edge detection on input image in x, y direction
    """
    return binary_output
```

首先我们需要将彩色图像转换为灰度图像。对于灰度图像，每个像素由uint8数字(0到255)表示。图像梯度可以很容易地强调边缘，因此将图像中的对象从背景中分离出来。

图6:图像梯度。*左:*原图。*右:*渐变图像。

然后我们需要得到x轴和y轴上的梯度。回想一下，我们可以用索贝尔算子来近似一阶导数。梯度是索贝尔算子和原始图像之间二维卷积的结果。对于这一部分，你会发现OpenCV中的*索贝尔*函数很有用。

$$\nabla(f) = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}] \tag{1}$$

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \; S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2}$$

$$G_x = S_x * I, \; G_y = S_y * I \tag{3}$$

最后我们需要将每个像素转换为unit8，然后应用阈值得到二值图像。如果一个像素的值在最小阈值和最大阈值之间，我们将该像素的值设置为1。否则设置为0。得到的二值图像将与颜色阈值函数的结果相结合，并传递给透视变换。

4.2 颜色阈值

在本节中，我们将实现一个函数，该函数使用颜色阈值来检测图像中有趣的特征(例如车道)。输入图像来自回调函数。输出应为突出白色和黄色的二值图像。

```
def color_threshold(self, img, thresh= (100, 255)):
    """
    Convert RGB to HSL and threshold to binary image using S channel
    """

    return binary_output
```

除了梯度法，我们还可以利用美国的车道标记通常是白色或黄色的信息。你可以只过滤掉白色和黄色以外的像素，剩下的很可能是车道上的。你可以使用不同的色彩空间(RGB, LAB, HSV, HSL, YUV⋯)，不同的通道和比较效果。有些人在车道检测中更喜欢RGB和HSL。可以随意探索其他色彩空间。

你需要首先将回调函数的输入图像转换为所需的颜色空间。然后在某些通道上应用阈值来过滤像素。在最终的二值图像中，这些像素应设置为1，其余像素应设置为0。

图7:颜色阈值。*左:*原图。*右:*彩色阈值图像。

4.3 视角转换

在本节中，我们将实现一个函数，将图像转换为*鸟瞰图*(从顶部向下看)。这将在2D平面上给出车道的几何表示，以及车辆在这些车道之间的相对位置和航向。这种视角在控制车辆时比第一人称视角更有用。在*鸟瞰图*中，图像上的像素距离将与实际距离成正比，从而简化了未来实验室中控制模块的计算。

输入来自来自颜色和梯度阈值的组合二值图像。输出是转换为*鸟瞰图*的图像。M和Minv分别是变换矩阵和逆变换矩阵。不用担心它们。

```
def perspective_transform(self, img, verbose= False):
    """
    Get bird's eye view from input image
    """

    return warped_ img, M, Minv
```

在透视变换期间，我们希望保持共线性(即，最初位于直线上的所有点在变换后仍然位于直线上)。透视变换需要一个3乘3的变换矩阵。

图8:透视变换。*左:*原图。*右:*鸟瞰图。

这里(x, y)和(u, v)是原视角和新视角坐标系中同一点的坐标。

$$\begin{bmatrix} tu \\ tv \\ t \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4}$$

为了找到矩阵，我们需要在原始图像上找到4个点的位置，并将相同的4个点映射到*Bird 's Eye View上*。这4个点中的任何3个都不应该在同一条线上。将这两组点放入cv2.getPerspectiveTransform()中，输出的将是变换矩阵。将矩阵传递到cv2.warpPerspective()中，我们将得到扭曲的*鸟瞰眼视图*图像。

4.4 巷配件

从上一步中，我们得到了二进制的*鸟瞰图*，它将可能的车道像素和背景分开。我们将图像水平切割成几层，并尝试在每一层上找到车道中心。这一步使用的是直方图方法。我们计算该水平层左半部分和右半部分像素的直方图。可能像素最密集的地方将被视为每条车道的质心。最后使用np.polyfit()函数将这些车道中心的坐标拟合为二阶多项式。增强可以通过取N个最近的帧多项式系数的平均值来实现。如果车道拟合失败，意味着多项式无法求解，程序将打印错误信息:*无法检测车道*。

5 检测车道线

5.1 使用rosbag的视频

ROS[2]中包含图像、传感器数据等的消息可以记录在*rosbag文件中*。这很方便，因为稍后可以重放rosbag来测试和调试我们将创建的软件模块。在一个rosbag中，来自不同来源的数据由不同的*rostopics组织*。

在对*rosbag进行任何操作之前*，我们需要启动ROS主节点:

```
roscore
```

对于这个MP，我们的输入图像由rostopic *camera/image_raw发布*。我们可以使用命令rosbag info和rosbag play来回放它。

我们在两个位置存储了4个rosbag，仓库中的bags目录(0011_sync。包,0056 _sync。bag，和0484_sync.bag)和一个更大的rosbag供你**从下面的盒子链接这里下载**。使用您的伊利诺斯州凭证访问这个，并将其与其他3个rosbag下载到文件夹中。你可以通过以下方式进入该目录:

```
cd [path-to-MP1-workspace]/src/mp1/bags
```

首先，在bag files目录中执行以下命令，查看rosbag中的内容类型。

```
rosbag info <your bagfile>
```

你应该会看到类似这样的内容:

```
path:     0011_sync.bag
version:  2.0
duration:  7.3s
start:    Dec 31 1969 18:00:00.00 (0.00)
end:      Dec 31 1969 18:00:07.30 (7.30)
size:     98.6 MB
messages:  74
compression: none [74/74 chunks]
types:    sensor_msgs/Image [060021388200f6f0f447d0fcd9c64743]
topics:   camera/image_raw  74 msgs  : sensor_msgs/Image
```

这告诉我们主题名称和类型以及包文件中包含的每个消息主题的数量(计数)。

下一步是重播包文件。在终端窗口中，在原始bag文件所在目录下运行以下命令:

```
rosbag play -l <your bagfile>
```

在播放rosbag时，视频将被转换成单独的图像，并以一定的频率发送出去。我们提供了一个回调函数。每次发布新图像时，都会触发回调函数，将图像从ROS消息格式转换为OpenCV格式，并将图像传递到我们的管道中。

5.2 GEM模型在凉亭

对于这个MP，我们还在Gazebo中提供了一个模拟环境，其中配备摄像机的GEM汽车沿着赛道移动。

编译和来源包

```
Python3−m PIP install scikit−image
源/ opt / ros /智力的设置。bash #您可以选择将此行添加到您的bashrc以方便您自己。catkin_make

源猛击/ setup.bash
```

要启动Gazebo，你需要执行以下命令:

```
roslaunch mp1 mp1.launch
```

To drive the vehicle along the track, execute python script:

```
python3 main.py
```

To reset the vehicle to original position, execute python script:

```
python3 reset.py
```

5.3 测试结果

代码完成后，只需运行:

```
python3 studentVision.py
```

Rviz是一个工具，可以帮助我们可视化模块输入和输出的ROS消息。在安装ROS时默认包含它。要启动它只需运行:

```
rviz
```

你可以在左下角找到一个“添加”按钮。按主题点击“添加->”。从主题列表中，选择“/车道检测-> /标注图像-> / image”(如图9所示)，应该会弹出一个小窗口，显示左右线已经叠加在原始视频上。重复上述程序，显示“/车道检测-> /Birdseye -> /Image”。结果应该如图10所示。

6 报告

每个小组都应上传一份简短的报告，并回答以下问题。

问题5[组](15分)在车道检测模块中创建不同的功能时，你考虑并执行了哪些有趣的设计选择?例如，你使用了哪些颜色空间?你如何确定透视变换的源点和目标点?请提供至少两个，并且是描述性的。一个简单的设计选择“我们写了一个脚本来选择参数”是不够的——脚本如何选择参数更相关。

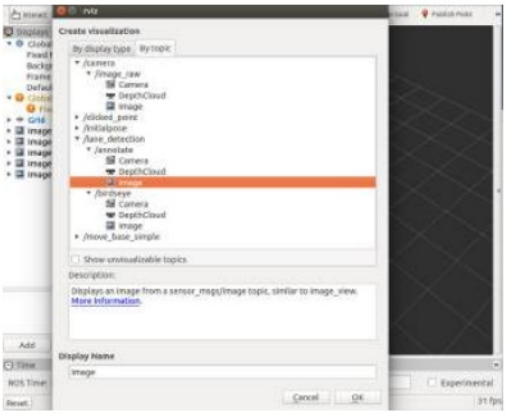

图9:选择主题

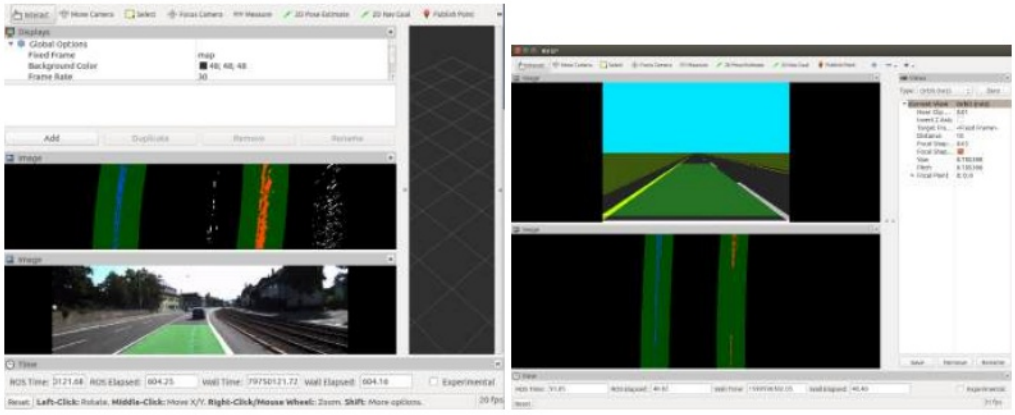

图10:Rviz的结果;左为rosbag场景，右为GEM car场景

问题6[组](25个点)为了检测两种场景中的车道线，您需要修改一些参数。请列出并比较你修改过的**所有**参数，并解释为什么修改这些参数有帮助?

问题7[组](30分)记录3段Rviz窗口和Gazebo的短视频，以表明你的代码适用于几种场景。你可以使用屏幕录制软件或智能手机进行录制。请确保您的视频链接在提交前是可用的。

问题8[组](20分)提供的rosbags (0484_sync.bag)中有一个是在降雪条件下记录的。你的车道探测器在尝试适应这种特定情况下的车道时可能会遇到困难。如果你的车道检测器有效，请报告你使用了什么技术来实现。如果不行，试着找出可能的原因并加以解释。(注意，在这种情况下，不一定会根据你的模型是否适合车道来评估你;相反，我们将根据您提供的推理进行评估。)

演示(10分)你需要在实验室演示期间向助教演示你在这两种情况下的解决方案。可能会有一个自动评分组件，使用你的解决方案和黄金解决方案之间的误差度量计算。

7 提交指令

问题1-4必须单独完成(作业1)。将每个问题的答案写在名为hw1_<netid>.pdf的文件中，并将文件上传到Canvas中。在pdf文件中包含你的名字和netid。你可以与他人讨论解决方案，但不要使用这些讨论中的笔记来写下你的答案。如果你使用/阅读任何本课程以外的材料来帮助解决问题，你应该明确地引用它们。

问题5-8可以以2-4人为一组完成。学生需要拍摄2-3个短视频，清楚地表明他们的代码在两种场景下都能工作(rosbag视频和模拟器环境)，并回答其他问题。请注意，对于模拟环境，您的视频必须显示您的实现可以在汽车**移动时检测车道**。视频可以上传到一个网站(YouTube, Vimeo，或Google Drive与公共访问)，你需要在报告中包括链接。每个团队的一名成员将报告mp1_<groupname>.pdf上传到Canvas。请包括所有小组成员的姓名和netids，并列举你在解决方案中可能使用的任何外部资源。此外，请将您的代码上传到一些云存储，如Google Drive，DropBox或Box;创建一个可分享的链接，并将其包含在你的报告中。请只上传包含除rosbags之外的代码的。zip文件中的src文件夹。

参考文献

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.